WHAT IS CLAIMED:

1. A method of verifying program code conversion performed by an emulator, comprising the step of:

a) executing subject code through an emulator on a subject processor up until a comparable point in the subject code;

b) executing the subject code natively on the subject processor up until the same comparable point in the subject code; and

c) comparing execution of the subject code natively on the subject processor against execution of the subject code on the subject processor through the emulator at the comparable point in the subject code.

2. The method of claim 1, wherein:

the step (a)comprises executing the subject code on the subject processor up until the comparable point in the subject code through the emulator to provide an emulated machine state;

the step (b) comprises executing the subject code natively on the subject processor up until the same comparable point in the subject code to provide a native machine state; and

the step (c) comprises comparing the emulated machine state against the native machine state at every comparable point in the subject code.

3. The method of claim 2, comprising performing the step (a) prior to performing the step (b).

4. The method of claim 3, wherein:

the step (a) comprises providing an emulated image of the subject processor and/or an emulated image of a memory associated with the subject processor;

the step (b) comprises providing a native image of the subject processor following the native execution of the program code and/or a native image of the memory associated with the subject processor, following the native execution of the program code; and

the step (c) comprises comparing the emulated image of the subject processor against the native image of the subject processor and/or comparing the emulated image of the memory against the native image of the memory.

5. The method of claim 4, wherein the step (a) comprises providing the emulated image of the memory in a load/store buffer associated with the memory, such that the memory is not affected by executing the subject code through the emulator.

6. The method of claim 5, wherein the emulated image of the subject processor includes an image of one or more registers.

7. The method of claim 6, wherein the emulated image of the subject processor includes an image of one or more condition code flags.

8. The method of claim 1, comprising executing the subject code natively and through the emulator both within a single process image of the subject processor.

9. The method of claim 8, comprising the step of performing a context switch between at least an emulation context for execution of the subject code through the emulator, and a native context for execution of the subject code natively on the subject processor, the native context and the emulation context being contexts within the single process image.

10. The method of claim 9, comprising selectively switching between an emulation context for running the emulator on the subject processor, a target execution context for executing target code produced by the emulator on the subject processor, and a subject native context where the subject code runs natively in the subject processor.

11. The method of claim 10, wherein both the native context and the emulation context employ a single image of the subject code.

12. The method of claim 2, comprising:

dividing the subject code into a plurality of blocks, wherein each block comprises one of the comparable points in the subject code,

executing one of the blocks, and

comparing machine states resulting from execution of the one block.

13. The method of claim 12, comprising selecting between two or more verification modes, and dividing the subject code into the plurality of blocks according to the selected verification mode.

14. The method of claim 13, comprising dividing the subject code into a plurality of blocks, and repeating the executing and comparing steps for each of the plurality of blocks.

15. The method of claim14, wherein each block comprises any one of:

(a) a single instruction of subject code;

(b) a basic block comprising a sequence of instructions from a unique entry instruction to a unique exit instruction; or

(c) a group block comprising a plurality of the basic blocks.

16. The method of claim 1, comprising the steps of:

dividing a large segment of the subject code into a plurality of smaller blocks, each block containing one or more instructions from the large segment of subject code; and

performing a verification comparison at a block boundary between each pair of consecutive neighbouring blocks in the plurality of blocks.

17. The method of claim 16, comprising the steps of:

providing the subject processor in an emulation context, where control of the processor rests with the emulator, performing program code conversion on a current

block BBn to produce a corresponding block of converted target code, and patching an immediately preceding block of subject code BBn-1 with a return jump;

executing a context switch routine to enter a subject native context, and executing the immediately preceding block of subject code BBn-1 natively by the subject processor, such that the executing step terminates with the return jump;

executing a context switch routine to return to the emulation context, and performing the verification comparison by comparing a native machine state representing the subject processor following execution of the immediately preceding block BBn-1 with an emulated machine state representing a virtual model of the subject processor held by the emulator following execution of the immediately preceding block BBn-1;

executing a context switch to a target execution context, and modelling execution of the target code corresponding to the current block of subject code BBn in the virtual model of the subject processor held by the emulator, thereby leaving the virtual model in a machine state representing the end of the current block BBn; and

repeating the above steps for each subsequent block in the plurality of blocks, unless the verification comparison reveals an error in the program code conversion.

18. The method of claim 17, further comprising restoring the immediately preceding block BBn-1 to remove the return jump.

19. The method of claim 1, further comprising the steps of:

selecting a block of the subject code;

executing the block of subject code on the subject processor through the emulator; and

appending a return jump to the block of subject code, and executing the block of subject code natively on the subject processor terminating with the return jump, such that the return jump returns control of the processor to the emulator.

20. A method of verifying program code conversion, comprising the steps of:

performing program code conversion to convert subject code into target code through an emulator running on a subject processor and executing the target code to provide an emulated machine state that is stored in a load/store buffer associated with the subject processor;

executing the subject code directly on the subject processor to provide a native machine state that is stored in a memory associated with the subject processor; and

comparing the emulated machine state contained in the load/store buffer against the native machine state contained in the memory to verify the program code conversion.

21. The method of claim 20, further comprising:

selectively inhibiting access by the emulator to the memory associated with the subject processor by buffering load and store requests from the subject processor to the memory in a load/store buffer.

22. The method of claim 21, comprising selectively inhibiting access to the memory when executing the target code, such that an emulated memory image is provided in the load/store buffer.

23. The method of claim 20, wherein once the program code conversion performed by the emulator running on the subject processor has been verified, the method further comprising the step of:

comparing execution of the subject code through the emulator running on the subject processor against execution of the subject code through a second emulator running on a target processor.

24. The method of claim 23, comprising providing a first host processor as the subject processor, and providing a second host processor as the target processor.

25. The method of claim 24, wherein the subject code is natively executable on the subject processor whilst not being natively executable on the target processor.

26. A method of verifying program code conversion, comprising the steps of:

first comparing execution of subject code natively on a subject processor against execution of the subject code on the subject processor through a first emulator, thereby verifying program code conversion performed by the first emulator; and

once program code conversion performed by the first emulator is verified, next comparing execution of subject code through the first emulator running on the subject processor against execution of the subject code through a second emulator running on a target processor, thereby verifying program code conversion performed by the second emulator using the verified program code conversion performed by the first emulator.

27. The method of claim 26, comprising the steps of:

performing a first program code conversion of the subject code including providing a first virtual model of the subject processor in the first emulator, and comparing the first virtual model against the subject processor; and

performing a second program code conversion of the subject code including providing a second virtual model of the subject processor in the second emulator, and comparing the first virtual model in the first emulator against the second virtual model in the second emulator.

28. The method of claim 27, comprising providing a single way communication from the first emulator to the second emulator.

29. The method of claim 27, comprising the steps of:

synchronising the first and second virtual models by sending initial state information from the first emulator to the second emulator;

dividing the subject code into a plurality of blocks;

for each block of subject code, executing the block of subject code through the first emulator and providing a set of subject machine state data and non-deterministic values to the second emulator;

executing the block of subject code in the second emulator substituting the non-deterministic values and providing a set of target machine state data; and

comparing the subject machine state data against the target machine state data and reporting an error if a divergence is detected, otherwise repeating the process for a next block of subject code.

30. A method of verifying program code conversion, comprising the steps of:

(a) dividing subject code into a plurality of blocks, wherein each block includes at least one instruction,

(b) executing one the blocks of subject code on a subject processor through a first emulator;

(c) comparing execution of the one block of subject code natively on a subject processor against the execution of the one block of subject code on the subject processor through the first emulator, thereby verifying program code conversion of the block of subject code performed by the first emulator;

(d) comparing execution of the same one block of subject code through a second emulator running on a target processor against the already verified execution of the one block of subject code through the first emulator running on the subject processor, thereby verifying program code conversion of the one block of subject code performed by the second emulator; and.

(e) repeating steps (b) – (d) for every block of the subject code until program code conversion performed by the second emulator is verified for every block of the subject code.

31. The method of claim 30, wherein the subject code is initially divided such that each block of subject code contains a single instruction.

32. The method of claim 31, wherein after program code conversion performed by the second emulator is verified for every block of subject code containing a single instruction, the method further comprises:

repeating step (a) by redividing the subject code into a plurality of new blocks, wherein each new block is a basic block comprising a sequence of instructions from a unique entry instruction to a unique exit instruction; and

repeating steps (b) – (e) for each basic block, thereby verifying program code conversion performed by the second emulator for every basic block of subject code.

33. The method of claim 32, wherein after program code conversion performed by the second emulator is verified for every basic block of subject code, the method further comprises:

repeating steps (a) by redividing the subject code into a plurality of group blocks, wherein each group block comprises a plurality of basic blocks; and

repeating steps (b) – (e) for each group block, thereby verifying program code conversion performed by the second emulator for every group block of subject code.

34. A computer-readable storage medium having emulator software resident thereon in the form of computer readable code executable by a computer for performing a method of verifying program code conversion performed by an emulator, the method comprising:

a) executing subject code through an emulator on a subject processor up until a comparable point in the subject code

b) executing the subject code natively on the subject processor up until the same comparable point in the subject code; and

c) comparing execution of the subject code natively on the subject processor against execution of the subject code on the subject processor through the emulator at the comparable point in the subject code.

35. The computer-readable storage medium of claim 34, wherein:

the step (a) comprises executing the subject code on the subject processor up until the comparable point in the subject code through the emulator to provide an emulated machine state;

the step (b) comprises executing the subject code natively on the subject processor up until the same comparable point in the subject code to provide a native machine state; and

the step (c) comprises comparing the emulated machine state against the native machine state at every comparable point in the subject code.

36. The computer-readable storage medium of claim 35, the method comprising performing the step (a) prior to performing the step (b).

37. The computer-readable storage medium of claim 36, wherein:

the step (a) comprises providing an emulated image of the subject processor and/or an emulated image of a memory associated with the subject processor;

the step (b) comprises providing a native image of the subject processor following the native execution of the program code and/or a native image of the memory associated with the subject processor, following the native execution of the program code; and

the step (c) comprises comparing the emulated image of the subject processor against the native image of the subject processor and/or comparing the emulated image of the memory against the native image of the memory.

38. The computer-readable storage medium of claim 37, wherein the step (a) comprises providing the emulated image of the memory in a load/store buffer associated with the memory, such that the memory is not affected by executing the subject code through the emulator.

39. The computer-readable storage medium of claim 38, wherein the emulated image of the subject processor includes an image of one or more registers.

40. The computer-readable storage medium of claim 39, wherein the emulated image of the subject processor includes an image of one or more condition code flags.

41. The computer-readable storage medium of claim 34, the method further comprising executing the subject code natively and through the emulator both within a single process image of the subject processor.

42. The computer-readable storage medium of claim 41, the method further comprising the step of performing a context switch between at least an emulation context for execution of the subject code through the emulator, and a native context for execution of the subject code natively on the subject processor, the native context and the emulation context being contexts within the single process image.

43. The computer-readable storage medium of claim 42, the method further comprising selectively switching between an emulation context for running the emulator on the subject processor, a target execution context for executing target code produced by the emulator on the subject processor, and a subject native context where the subject code runs natively in the subject processor.

44. The computer-readable storage medium of claim 43, wherein both the native context and the emulation context employ a single image of the subject code.

45. The computer-readable storage medium of claim 35, the method further comprising:

dividing the subject code into a plurality of blocks, wherein each block comprises one of the comparable points in the subject code,

executing one of the blocks, and

comparing machine states resulting from execution of the one block.

46. The computer-readable storage medium of claim 45, the method further comprising selecting between two or more verification modes, and dividing the subject code into the plurality of blocks according to the selected verification mode.

47. The computer-readable storage medium of claim 46, the method further comprising dividing the subject code into a plurality of blocks, and repeating the executing and comparing steps for each of the plurality of blocks.

48. The computer-readable storage medium of claim 47, wherein each block comprises any one of:

(a) a single instruction of subject code;

(b) a basic block comprising a sequence of instructions from a unique entry instruction to a unique exit instruction; or

(c) a group block comprising a plurality of the basic blocks.

49. The computer-readable storage medium of claim 34, the method further comprising the steps of:

dividing a large segment of the subject code into a plurality of smaller blocks, each block containing one or more instructions from the large segment of subject code; and

performing a verification comparison at a block boundary between each pair of consecutive neighbouring blocks in the plurality of blocks.

50. The computer-readable storage medium of claim 49, the method further comprising the steps of:

providing the subject processor in an emulation context, where control of the processor rests with the emulator, performing program code conversion on a current block BBn to produce a corresponding block of converted target code, and patching an immediately preceding block of subject code BBn-1 with a return jump;

executing a context switch routine to enter a subject native context, and executing the immediately preceding block of subject code BBn-1 natively by the subject processor, such that the executing step terminates with the return jump;

executing a context switch routine to return to the emulation context, and performing the verification comparison by comparing a native machine state representing the subject processor following execution of the immediately preceding

block BBn-1 with an emulated machine state representing a virtual model of the subject processor held by the emulator following execution of the immediately preceding block BBn-1;

executing a context switch to a target execution context, and modelling execution of the target code corresponding to the current block of subject code BBn in the virtual model of the subject processor held by the emulator, thereby leaving the virtual model in a machine state representing the end of the current block BBn; and

repeating the above steps for each subsequent block in the plurality of blocks, unless the verification comparison reveals an error in the program code conversion.

51. The computer-readable storage medium of claim 50, the method further comprising restoring the immediately preceding block BBn-1 to remove the return jump.

52. The computer-readable storage medium of claim 34, the method further comprising the steps of:

selecting a block of the subject code;

executing the block of subject code on the subject processor through the emulator; and

appending a return jump to the block of subject code, and executing the block of subject code natively on the subject processor terminating with the return jump, such that the return jump returns control of the processor to the emulator.

53. A computer-readable storage medium having emulator software resident thereon in the form of computer readable code executable by a computer for performing a method of verifying program code conversion performed by an emulator, the method comprising:

performing program code conversion to convert subject code into target code through an emulator running on a subject processor and executing the target code to provide an emulated machine state that is stored in a load/store buffer associated with the subject processor;

executing the subject code directly on the subject processor to provide a native machine state that is stored in a memory associated with the subject processor; and

comparing the emulated machine state contained in the load/store buffer against the native machine state contained in the memory to verify the program code conversion.

54. The computer-readable storage medium of claim 53, the method further comprising:

selectively inhibiting access by the emulator to the memory associated with the subject processor by buffering load and store requests from the subject processor to the memory in a load/store buffer.

55. The computer-readable storage medium of claim 54, the method further comprising selectively inhibiting access to the memory when executing the target code, such that an emulated memory image is provided in the load/store buffer.

56. The computer-readable storage medium of claim 53, wherein once the program code conversion performed by the emulator running on the subject processor has been verified, the method further comprising the step of:

comparing execution of the subject code through the emulator running on the subject processor against execution of the subject code through a second emulator running on a target processor.

57. The computer-readable storage medium of claim 56, the method further comprising providing a first host processor as the subject processor, and providing a second host processor as the target processor.

58. The computer-readable storage medium of claim 57, wherein the subject code is natively executable on the subject processor whilst not being natively executable on the target processor.

59. A computer-readable storage medium having emulator software resident thereon in the form of computer readable code executable by a computer for performing a method of verifying program code conversion performed by an emulator, the method comprising:

first comparing execution of subject code natively on a subject processor against execution of the subject code on the subject processor through a first emulator, thereby verifying program code conversion performed by the first emulator; and

once program code conversion performed by the first emulator is verified, next comparing execution of subject code through the first emulator running on the subject processor against execution of the subject code through a second emulator running on a target processor, thereby verifying program code conversion performed by the second emulator using the verified program code conversion performed by the first emulator.

60. The computer-readable storage medium of claim 59, the method further comprising the steps of:

performing a first program code conversion of the subject code including providing a first virtual model of the subject processor in the first emulator, and comparing the first virtual model against the subject processor; and

performing a second program code conversion of the subject code including providing a second virtual model of the subject processor in the second emulator, and comparing the first virtual model in the first emulator against the second virtual model in the second emulator.

61. The computer-readable storage medium of claim 60, the method further comprising providing a single way communication from the first emulator to the second emulator.

62. The computer-readable storage medium of claim 60, the method further comprising the steps of:

synchronising the first and second virtual models by sending initial state information from the first emulator to the second emulator;

dividing the subject code into a plurality of blocks;

for each block of subject code, executing the block of subject code through the first emulator and providing a set of subject machine state data and non-deterministic values to the second emulator;

executing the block of subject code in the second emulator substituting the non-deterministic values and providing a set of target machine state data; and

comparing the subject machine state data against the target machine state data and reporting an error if a divergence is detected, otherwise repeating the process for a next block of subject code.

63. A computer-readable storage medium having emulator software resident thereon in the form of computer readable code executable by a computer for performing a method of verifying program code conversion performed by an emulator, the method comprising:

(a) dividing subject code into a plurality of blocks, wherein each block includes at least one instruction,

(b) executing one the blocks of subject code on a subject processor through a first emulator;

(c) comparing execution of the one block of subject code natively on a subject processor against the execution of the one block of subject code on the subject processor through the first emulator, thereby verifying program code conversion of the block of subject code performed by the first emulator;

(d) comparing execution of the same one block of subject code through a second emulator running on a target processor against the already verified execution of the one block of subject code through the first emulator running on the subject processor, thereby verifying program code conversion of the one block of subject code performed by the second emulator; and.

(e) repeating steps (b) – (d) for every block of the subject code until program code conversion performed by the second emulator is verified for every block of the subject code.

64. The computer-readable storage medium of claim 63, wherein the subject code is initially divided such that each block of subject code contains a single instruction.

65. The computer-readable storage medium of claim 64, wherein after program code conversion performed by the second emulator is verified for every block of subject code containing a single instruction, the method further comprises:

repeating step (a) by redividing the subject code into a plurality of new blocks, wherein each new block is a basic block comprising a sequence of instructions from a unique entry instruction to a unique exit instruction;

repeating steps (b) – (e) for each basic block, thereby verifying program code conversion performed by the second emulator for every basic block of subject code.

66. The computer-readable storage medium of claim 65, wherein after program code conversion performed by the second emulator is verified for every basic block of subject code, the method further comprises:

repeating steps (a) by redividing the subject code into a plurality of group blocks, wherein each group block comprises a plurality of basic blocks; and

repeating steps (b) – (e) for each group block, thereby verifying program code conversion performed by the second emulator for every group block of subject code.

67. An emulator apparatus comprising in combination:

a processor; and

emulator code for performing a method of verifying program code conversion performed by an emulator, said emulator code comprising code executable by said processor for performing the following steps:

a) executing subject code through an emulator on a subject processor up until a comparable point in the subject code

b) executing the subject code natively on the subject processor up until the same comparable point in the subject code; and

c) comparing execution of the subject code natively on the subject processor against execution of the subject code on the subject processor through the emulator at the comparable point in the subject code.

68. The emulator apparatus of claim 67, wherein:

the step (a)comprises executing the subject code on the subject processor up until the comparable point in the subject code through the emulator to provide an emulated machine state;

the step (b) comprises executing the subject code natively on the subject processor up until the same comparable point in the subject code to provide a native machine state; and

the step (c) comprises comparing the emulated machine state against the native machine state at every comparable point in the subject code.

69. The emulator apparatus of claim 68, comprising performing the step (a) prior to performing the step (b).

70. The emulator apparatus of claim 69, wherein:

the step (a) comprises providing an emulated image of the subject processor and/or an emulated image of a memory associated with the subject processor;

the step (b) comprises providing a native image of the subject processor following the native execution of the program code and/or a native image of the memory associated with the subject processor, following the native execution of the program code; and

the step (c) comprises comparing the emulated image of the subject processor against the native image of the subject processor and/or comparing the emulated image of the memory against the native image of the memory.

71. The emulator apparatus of claim 70, wherein the step (a) comprises providing the emulated image of the memory in a load/store buffer associated with the memory, such that the memory is not affected by executing the subject code through the emulator.

72. The emulator apparatus of claim 71, wherein the emulated image of the subject processor includes an image of one or more registers.

73. The emulator apparatus of claim 72, wherein the emulated image of the subject processor includes an image of one or more condition code flags.

74. The emulator apparatus of claim 67, said emulator code further comprising code executable by said processor for executing the subject code natively and through the emulator both within a single process image of the subject processor.

75. The emulator apparatus of claim 74, said emulator code further comprising code executable by said processor for performing a context switch between at least an emulation context for execution of the subject code through the emulator, and a native context for execution of the subject code natively on the subject processor, the native context and the emulation context being contexts within the single process image.

76. The emulator apparatus of claim 75, said emulator code further comprising code executable by said processor for selectively switching between an emulation context for running the emulator on the subject processor, a target execution context for executing target code produced by the emulator on the subject processor, and a subject native context where the subject code runs natively in the subject processor.

77. The emulator apparatus of claim 76, wherein both the native context and the emulation context employ a single image of the subject code.

78. The emulator apparatus of claim 68, said emulator code further comprising code executable by said processor for performing the following steps:

dividing the subject code into a plurality of blocks, wherein each block comprises one of the comparable points in the subject code,

executing one of the blocks, and

comparing machine states resulting from execution of the one block.

79. The emulator apparatus of claim 78, said emulator code further comprising code executable by said processor for selecting between two or more verification modes, and dividing the subject code into the plurality of blocks according to the selected verification mode.

80. The emulator apparatus of claim 79, said emulator code further comprising code executable by said processor for dividing the subject code into a plurality of blocks, and repeating the executing and comparing steps for each of the plurality of blocks.

81. The emulator apparatus of claim 80, wherein each block comprises any one of:

(a) a single instruction of subject code;

(b) a basic block comprising a sequence of instructions from a unique entry instruction to a unique exit instruction; or

(c) a group block comprising a plurality of the basic blocks.

82. The emulator apparatus of claim 67, said emulator code further comprising code executable by said processor for performing the steps of:

dividing a large segment of the subject code into a plurality of smaller blocks, each block containing one or more instructions from the large segment of subject code; and

performing a verification comparison at a block boundary between each pair of consecutive neighbouring blocks in the plurality of blocks.

83. The emulator apparatus of claim 82, said emulator code further comprising code executable by said processor for performing the steps of:

providing the subject processor in an emulation context, where control of the processor rests with the emulator, performing program code conversion on a current block BBn to produce a corresponding block of converted target code, and patching an immediately preceding block of subject code BBn-1 with a return jump;

executing a context switch routine to enter a subject native context, and executing the immediately preceding block of subject code BBn-1 natively by the subject processor, such that the executing step terminates with the return jump;

executing a context switch routine to return to the emulation context, and performing the verification comparison by comparing a native machine state representing the subject processor following execution of the immediately preceding block BBn-1 with an emulated machine state representing a virtual model of the subject processor held by the emulator following execution of the immediately preceding block BBn-1;

executing a context switch to a target execution context, and modelling execution of the target code corresponding to the current block of subject code BBn in the virtual model of the subject processor held by the emulator, thereby leaving the virtual model in a machine state representing the end of the current block BBn; and

repeating the above steps for each subsequent block in the plurality of blocks, unless the verification comparison reveals an error in the program code conversion.

84. The emulator apparatus of claim 83, said emulator code further comprising code executable by said processor for restoring the immediately preceding block BBn-1 to remove the return jump.

85. The emulator apparatus of claim 67, said emulator code further comprising code executable by said processor for performing the steps of:

selecting a block of the subject code;

executing the block of subject code on the subject processor through the emulator; and

appending a return jump to the block of subject code, and executing the block of subject code natively on the subject processor terminating with the return jump, such that the return jump returns control of the processor to the emulator.

86. An emulator apparatus comprising in combination:

a processor; and

emulator code for performing a method of verifying program code conversion performed by an emulator, said emulator code comprising code executable by said processor for performing the following steps:

performing program code conversion to convert subject code into target code through an emulator running on a subject processor and executing the target code to provide an emulated machine state that is stored in a load/store buffer associated with the subject processor;

executing the subject code directly on the subject processor to provide a native machine state that is stored in a memory associated with the subject processor; and

comparing the emulated machine state contained in the load/store buffer against the native machine state contained in the memory to verify the program code conversion.

87. The emulator apparatus of claim 86, said emulator code further comprising code executable by said processor for selectively inhibiting access by the emulator to the memory associated with the subject processor by buffering load and store requests from the subject processor to the memory in a load/store buffer.

88. The emulator apparatus of claim 87, said emulator code further comprising code executable by said processor for selectively inhibiting access to the memory when executing the target code, such that an emulated memory image is provided in the load/store buffer.

89. The emulator apparatus of claim 86, once the program code conversion performed by the emulator running on the subject processor has been verified, said emulator code further comprising code executable by said processor for comparing execution of the subject code through the emulator running on the subject processor against execution of the subject code through a second emulator running on a target processor.

90. The emulator apparatus of claim 89, said emulator code further comprising code executable by said processor for providing a first host processor as the subject processor, and providing a second host processor as the target processor.

91. The emulator apparatus of claim 90, wherein the subject code is natively executable on the subject processor whilst not being natively executable on the target processor.

92. An emulator apparatus comprising in combination:

a processor; and

emulator code for performing a method of verifying program code conversion performed by an emulator, said emulator code comprising code executable by said processor for performing the following steps:

first comparing execution of subject code natively on a subject processor against execution of the subject code on the subject processor through a first emulator, thereby verifying program code conversion performed by the first emulator; and

once program code conversion performed by the first emulator is verified, next comparing execution of subject code through the first emulator running on the subject processor against execution of the subject code through a second emulator running on a target processor, thereby verifying program code conversion performed by the second emulator using the verified program code conversion performed by the first emulator.

93. The emulator apparatus of claim 92, said emulator code further comprising code executable by said processor for performing the following steps:

performing a first program code conversion of the subject code including providing a first virtual model of the subject processor in the first emulator, and comparing the first virtual model against the subject processor; and

performing a second program code conversion of the subject code including providing a second virtual model of the subject processor in the second emulator, and

comparing the first virtual model in the first emulator against the second virtual model in the second emulator.

94. The emulator apparatus of claim 93, said emulator code further comprising code executable by said processor for providing a single way communication from the first emulator to the second emulator.

95. The emulator apparatus of claim 93, said emulator code further comprising code executable by said processor for performing the following steps:

synchronising the first and second virtual models by sending initial state information from the first emulator to the second emulator;

dividing the subject code into a plurality of blocks;

for each block of subject code, executing the block of subject code through the first emulator and providing a set of subject machine state data and non-deterministic values to the second emulator;

executing the block of subject code in the second emulator substituting the non-deterministic values and providing a set of target machine state data; and

comparing the subject machine state data against the target machine state data and reporting an error if a divergence is detected, otherwise repeating the process for a next block of subject code.

96. An emulator apparatus comprising in combination:

a processor; and

emulator code for performing a method of verifying program code conversion performed by an emulator, said emulator code comprising code executable by said processor for performing the following steps:

(a) dividing subject code into a plurality of blocks, wherein each block includes at least one instruction;

(b) executing one the blocks of subject code on a subject processor through a first emulator;

(c) comparing execution of the one block of subject code natively on a subject processor against the execution of the one block of subject code on the subject

processor through the first emulator, thereby verifying program code conversion of the block of subject code performed by the first emulator;

(d) comparing execution of the same one block of subject code through a second emulator running on a target processor against the already verified execution of the one block of subject code through the first emulator running on the subject processor, thereby verifying program code conversion of the one block of subject code performed by the second emulator; and.

(e) repeating steps (b) – (d) for every block of the subject code until program code conversion performed by the second emulator is verified for every block of the subject code.

97. The emulator apparatus of claim 96, wherein the subject code is initially divided such that each block of subject code contains a single instruction.

98. The emulator apparatus of claim 97, wherein after program code conversion performed by the second emulator is verified for every block of subject code containing a single instruction, said emulator code further comprising code executable by said processor for performing the following steps:

repeating step (a) by redividing the subject code into a plurality of new blocks, wherein each new block is a basic block comprising a sequence of instructions from a unique entry instruction to a unique exit instruction; and

repeating steps (b) – (e) for each basic block, thereby verifying program code conversion performed by the second emulator for every basic block of subject code.

99. The emulator apparatus of claim 98, wherein after program code conversion performed by the second emulator is verified for every basic block of subject code, said emulator code further comprising code executable by said processor for performing the following steps:

repeating steps (a) by redividing the subject code into a plurality of group blocks, wherein each group block comprises a plurality of basic blocks; and

repeating steps (b) – (e) for each group block, thereby verifying program code conversion performed by the second emulator for every group block of subject code.